```python
import gdown
import pandas as pd
from sentence_transformers import SentenceTransformer
from sklearn.preprocessing import StandardScaler

customerCsv : str = "https://drive.google.com/file/d/1bu_--mo79VdUG9oin4ybfFGRUSXAe-WE/view?usp=sharing"
productsCsv : str = "https://drive.google.com/file/d/1IKuDizVapw-hyktwfpoAoaGtHtTNHfd0/view?usp=sharing"
transactionsCsv : str = "https://drive.google.com/file/d/1saEqdbBB-vuk2hxoAf4TzDEsykdKlzbF/view?usp=sharing"

def download_csv(url : str, outputPath : str) -> None:
  try:
    gdown.download(url, outputPath, quiet=False, fuzzy=True)
    print(f"File downloaded successfully and saved to {outputPath}")
  except Exception as e:
    print(f"Failed to dowload the file {e}")

download_csv(customerCsv, "/content/")
download_csv(productsCsv, "/content/")
download_csv(transactionsCsv, "/content/")
```

```
Downloading...
From: https://drive.google.com/uc?id=1bu_--mo79VdUG9oin4ybfFGRUSXAe-WE
To: /content/Customers.csv
100%|██████████| 8.54k/8.54k [00:00<00:00, 22.9MB/s]

File downloaded successfully and saved to /content/

Downloading...
From: https://drive.google.com/uc?id=1IKuDizVapw-hyktwfpoAoaGtHtTNHfd0
To: /content/Products.csv
100%|██████████| 4.25k/4.25k [00:00<00:00, 3.72MB/s]

File downloaded successfully and saved to /content/

Downloading...
From: https://drive.google.com/uc?id=1saEqdbBB-vuk2hxoAf4TzDEsykdKlzbF
To: /content/Transactions.csv
100%|██████████| 54.7k/54.7k [00:00<00:00, 70.4MB/s]

File downloaded successfully and saved to /content/
```

Preprocessing

```
dfCustomers = pd.read_csv("/content/Customers.csv")
dfTransactions = pd.read_csv("/content/Transactions.csv")
dfProducts = pd.read_csv("/content/Products.csv")

transactions_products = pd.merge(dfTransactions, dfProducts,
on='ProductID', how='left')
data = pd.merge(transactions_products, dfCustomers, on='CustomerID',
how='left')
data.head()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"TransactionID\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
1000,\n        \"samples\": [\n          \"T00677\",\n
\"T00790\",\n          \"T00907\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CustomerID\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 199,\n        \"samples\": [\n
\"C0135\",\n          \"C0109\",\n          \"C0048\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ProductID\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 100,\n        \"samples\": [\n
\"P082\",\n          \"P052\",\n          \"P035\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TransactionDate\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"2024-03-05 23:39:40\",\n          \"2024-08-13 23:52:47\",\n
\"2024-02-15 17:18:56\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Quantity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 4,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          4,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TotalValue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
493.14447754793144,\n        \"min\": 16.08,\n        \"max\":
1991.04,\n        \"num_unique_values\": 369,\n        \"samples\": [\
n          1789.36,\n          681.78,\n          580.34\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Price_x\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
140.73638962578207,\n        \"min\": 16.08,\n        \"max\":
497.76,\n        \"num_unique_values\": 100,\n        \"samples\": [\n
55.99,\n          354.81,\n          30.59\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"ProductName\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 66,\n   \"samples\": [\n
\"ActiveWear Jacket\",\n   \"BookWorld Bluetooth Speaker\",\n
\"ComfortLiving Bluetooth Speaker\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Category\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
4,\n   \"samples\": [\n   \"Clothing\",\n   \"Home
Decor\",\n   \"Electronics\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Price_y\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
140.73638962578207,\n   \"min\": 16.08,\n   \"max\":
497.76,\n   \"num_unique_values\": 100,\n   \"samples\": [\n
55.99,\n   354.81,\n   30.59\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"CustomerName\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 199,\n   \"samples\": [\n   \"Toni
Weaver\",\n   \"Abigail Jones\",\n   \"Matthew Park\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Region\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
4,\n   \"samples\": [\n   \"Asia\",\n   \"North
America\",\n   \"Europe\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"SignupDate\",\n
\"properties\": {\n   \"dtype\": \"object\",\n
\"num_unique_values\": 178,\n   \"samples\": [\n   \"2023-
06-11\",\n   \"2023-09-27\",\n   \"2022-02-10\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"data"}

```
data.info()

missing_values = data.isnull().sum()

duplicates = data.duplicated().sum()

missing_values, duplicates
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TransactionID    1000 non-null   object
 1   CustomerID       1000 non-null   object
```

```
 2   ProductID        1000 non-null   object
 3   TransactionDate  1000 non-null   object
 4   Quantity         1000 non-null   int64
 5   TotalValue       1000 non-null   float64
 6   Price_x          1000 non-null   float64
 7   ProductName      1000 non-null   object
 8   Category         1000 non-null   object
 9   Price_y          1000 non-null   float64
 10  CustomerName     1000 non-null   object
 11  Region           1000 non-null   object
 12  SignupDate       1000 non-null   object
dtypes: float64(3), int64(1), object(9)
memory usage: 101.7+ KB
```

```
(TransactionID      0
 CustomerID         0
 ProductID          0
 TransactionDate    0
 Quantity           0
 TotalValue         0
 Price_x            0
 ProductName        0
 Category           0
 Price_y            0
 CustomerName       0
 Region             0
 SignupDate         0
 dtype: int64,
 0)
```

```
data_cleaned = data.drop(columns=["Price_y"])


data_cleaned["TransactionDate"] =
pd.to_datetime(data_cleaned["TransactionDate"])
data_cleaned["SignupDate"] =
pd.to_datetime(data_cleaned["SignupDate"])


customer_profiles = data_cleaned.groupby("CustomerID").agg({
    "ProductName": lambda x: " ".join(x),
    "Category": lambda x: " ".join(x),
    "TotalValue": "sum",
    "Quantity": "sum",
    "TransactionDate": ["count", "max"],
    "SignupDate": "first",
    "Region": "first"
}).reset_index()
```

```
customer_profiles.columns = [
    "CustomerID", "ProductName", "Category", "TotalValue",
    "TotalQuantity", "TransactionCount", "LastTransactionDate",
    "SignupDate", "Region"
]


customer_profiles["DaysSinceSignup"] = (
    pd.Timestamp.now() - customer_profiles["SignupDate"]
).dt.days

customer_profiles["DaysSinceLastTransaction"] = (
    pd.Timestamp.now() - customer_profiles["LastTransactionDate"]
).dt.days

print(customer_profiles.head())
```

```
  CustomerID                                        ProductName  \
0      C0001  SoundWave Cookbook HomeSense Wall Art SoundWav...
1      C0002  BookWorld Cookware Set BookWorld Rug ComfortLi...
2      C0003  ActiveWear Cookware Set ActiveWear Rug ActiveW...
3      C0004  TechPro Textbook TechPro Rug TechPro Vase Acti...
4      C0005  ActiveWear Cookware Set TechPro Smartwatch Com...

                                        Category  TotalValue  \
0  Books Home Decor Electronics Electronics Elect...     3354.52
1            Home Decor Home Decor Clothing Clothing     1862.74
2          Home Decor Home Decor Clothing Electronics     2725.38
3  Books Home Decor Home Decor Home Decor Books B...     5354.88
4                  Home Decor Electronics Electronics     2034.24

   TotalQuantity  TransactionCount LastTransactionDate SignupDate  \
0             12                 5 2024-11-02 17:04:16 2022-07-10
1             10                 4 2024-12-03 01:41:41 2022-02-13
2             14                 4 2024-08-24 18:54:04 2024-03-07
3             23                 8 2024-12-23 14:13:52 2022-10-09
4              7                 3 2024-11-04 00:30:22 2022-08-15

          Region  DaysSinceSignup  DaysSinceLastTransaction
0  South America              929                        83
1           Asia             1076                        52
2  South America              323                       152
3  South America              838                        32
4           Asia              893                        81
```

```
customer_profiles["AveragePrice"] = customer_profiles["TotalValue"] /
customer_profiles["TotalQuantity"]

def categorize_price(price):
    if price < 100:
```

```
        return "Low"
    elif 100 <= price <= 500:
        return "Mid"
    else:
        return "High"

customer_profiles["PriceTier"] =
customer_profiles["AveragePrice"].apply(categorize_price)

price_tier_counts = customer_profiles.groupby(["CustomerID",
"PriceTier"]).size().unstack(fill_value=0)

for tier in ["Low", "Mid", "High"]:
    if tier not in price_tier_counts.columns:
        price_tier_counts[tier] = 0

price_tier_proportions =
price_tier_counts.div(price_tier_counts.sum(axis=1),
axis=0).reset_index()
price_tier_proportions.columns = ["CustomerID", "LowTierProp",
"MidTierProp", "HighTierProp"]

customer_profiles = customer_profiles.merge(price_tier_proportions,
on="CustomerID", how="left")

customer_profiles.drop(columns=["PriceTier"], inplace=True,
errors="ignore")

print(customer_profiles.head())
```

```
  CustomerID                                        ProductName  \
0      C0001  SoundWave Cookbook HomeSense Wall Art SoundWav...
1      C0002  BookWorld Cookware Set BookWorld Rug ComfortLi...
2      C0003  ActiveWear Cookware Set ActiveWear Rug ActiveW...
3      C0004  TechPro Textbook TechPro Rug TechPro Vase Acti...
4      C0005  ActiveWear Cookware Set TechPro Smartwatch Com...

                                          Category  TotalValue  \
0  Books Home Decor Electronics Electronics Elect...     3354.52
1            Home Decor Home Decor Clothing Clothing     1862.74
2         Home Decor Home Decor Clothing Electronics     2725.38
3  Books Home Decor Home Decor Home Decor Books B...     5354.88
4                Home Decor Electronics Electronics     2034.24

   TotalQuantity  TransactionCount LastTransactionDate SignupDate  \
0             12                 5 2024-11-02 17:04:16 2022-07-10
1             10                 4 2024-12-03 01:41:41 2022-02-13
2             14                 4 2024-08-24 18:54:04 2024-03-07
3             23                 8 2024-12-23 14:13:52 2022-10-09
4              7                 3 2024-11-04 00:30:22 2022-08-15
```

```
          Region  DaysSinceSignup  DaysSinceLastTransaction
AveragePrice  \
0  South America              929                        83
279.543333
1           Asia             1076                        52
186.274000
2  South America              323                       152
194.670000
3  South America              838                        32
232.820870
4           Asia              893                        81
290.605714

   LowTierProp  MidTierProp  HighTierProp
0          0.0          1.0           0.0
1          0.0          1.0           0.0
2          0.0          1.0           0.0
3          0.0          1.0           0.0
4          0.0          1.0           0.0
```

Feature Engineering

```
customer_profiles["TextFeatures"] = customer_profiles["ProductName"] +
" " + customer_profiles["Category"]

sbert_model = SentenceTransformer('all-MiniLM-L6-v2')

text_embeddings =
sbert_model.encode(customer_profiles["TextFeatures"].tolist(),
convert_to_tensor=False)

embedding_df = pd.DataFrame(text_embeddings, columns=[f"Embedding_{i}"
for i in range(len(text_embeddings[0]))])

customer_profiles = pd.concat([customer_profiles, embedding_df],
axis=1)

customer_profiles.to_csv("enhanced_customer_profiles_with_embeddings.c
sv", index=False)

print("Customer profiles updated with text embeddings and saved to
'enhanced_customer_profiles_with_embeddings.csv'.")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
```

```
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(

{"model_id":"d16351929c1342f5a722641f5faa8e31","version_major":2,"vers
ion_minor":0}

{"model_id":"0281038cd3984043a011537e55818238","version_major":2,"vers
ion_minor":0}

{"model_id":"ed144f514f604c39abd4205a1943d0fb","version_major":2,"vers
ion_minor":0}

{"model_id":"702d60347b4040e58efb1de1caf5369d","version_major":2,"vers
ion_minor":0}

{"model_id":"81fbc8fb7fa64b90b9f33d9dc245ad77","version_major":2,"vers
ion_minor":0}

{"model_id":"be73cf777ae946e58f604e41f49319e9","version_major":2,"vers
ion_minor":0}

{"model_id":"8d605bb2de65460ea9d12c8c2816646c","version_major":2,"vers
ion_minor":0}

{"model_id":"fd177722ded440149c7b375c767f03cb","version_major":2,"vers
ion_minor":0}

{"model_id":"83c8aa09d8b54303855f4b3b9bb8ec39","version_major":2,"vers
ion_minor":0}

{"model_id":"3a0e4591ea6f4179a35becd5b1b09fb0","version_major":2,"vers
ion_minor":0}

{"model_id":"4ea9761364f348c7873687c04833e164","version_major":2,"vers
ion_minor":0}

Customer profiles updated with text embeddings and saved to
'enhanced_customer_profiles_with_embeddings.csv'.
```

```
customer_profiles["PurchaseFrequency"] =
customer_profiles["TransactionCount"] /
customer_profiles["DaysSinceSignup"]
customer_profiles["AverageSpendPerTransaction"] =
customer_profiles["TotalValue"] /
customer_profiles["TransactionCount"]

customer_profiles["PurchaseFrequency"] =
customer_profiles["PurchaseFrequency"].fillna(0).replace([float("inf")
, -float("inf")], 0)
```

```
customer_profiles["AverageSpendPerTransaction"] =
customer_profiles["AverageSpendPerTransaction"].fillna(0).replace([flo
at("inf"), -float("inf")], 0)

numerical_features = [
    "TotalValue",
    "TotalQuantity",
    "DaysSinceSignup",
    "DaysSinceLastTransaction",
    "PurchaseFrequency",
    "AverageSpendPerTransaction",
]

scaler = StandardScaler()
customer_profiles[numerical_features] =
scaler.fit_transform(customer_profiles[numerical_features])

customer_profiles.to_csv("enhanced_customer_profiles_with_numerical_fe
atures.csv", index=False)

print("Numerical features engineered, standardized, and saved to
'enhanced_customer_profiles_with_numerical_features.csv'.")
```

```
Numerical features engineered, standardized, and saved to
'enhanced_customer_profiles_with_numerical_features.csv'.
```

Modelling

```
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
import pandas as pd

customer_profiles =
pd.read_csv("enhanced_customer_profiles_with_numerical_features.csv")

embedding_columns = [col for col in customer_profiles.columns if
col.startswith("Embedding_")]
numerical_features = [
    "TotalValue",
    "TotalQuantity",
    "DaysSinceSignup",
    "DaysSinceLastTransaction",
    "PurchaseFrequency",
    "AverageSpendPerTransaction",
]

text_similarity =
cosine_similarity(customer_profiles[embedding_columns])
```

```
numerical_data = customer_profiles[numerical_features].values
numerical_similarity = cosine_similarity(numerical_data)

TEXT_WEIGHT = 0.6
NUMERICAL_WEIGHT = 0.4
hybrid_similarity = (
    TEXT_WEIGHT * text_similarity + NUMERICAL_WEIGHT *
numerical_similarity
)

customer_ids = customer_profiles["CustomerID"]
recommendations = {}

for i, customer_id in enumerate(customer_ids[:20]):
    similarities = list(enumerate(hybrid_similarity[i]))
    similarities = sorted(similarities, key=lambda x: x[1],
reverse=True)

    top_3 = [(customer_ids[j], score) for j, score in similarities if
customer_ids[j] != customer_id][:3]
    recommendations[customer_id] = top_3

recommendations_df = pd.DataFrame({
    "CustomerID": recommendations.keys(),
    "Recommendations": [str(recommendations[cust]) for cust in
recommendations.keys()]
})

recommendations_df.to_csv("lookalike_recommendations.csv",
index=False)

print("Lookalike recommendations saved to
'lookalike_recommendations.csv'.")
```

```
Lookalike recommendations saved to 'lookalike_recommendations.csv'.
```

Recommendations

```
first_20_customers = customer_ids[:20]

recommendations_for_20 = {}

for i, customer_id in enumerate(first_20_customers):
    similarities = list(enumerate(hybrid_similarity[i]))
    similarities = sorted(similarities, key=lambda x: x[1],
reverse=True)

    top_3 = [(customer_ids[j], round(score, 4)) for j, score in
```

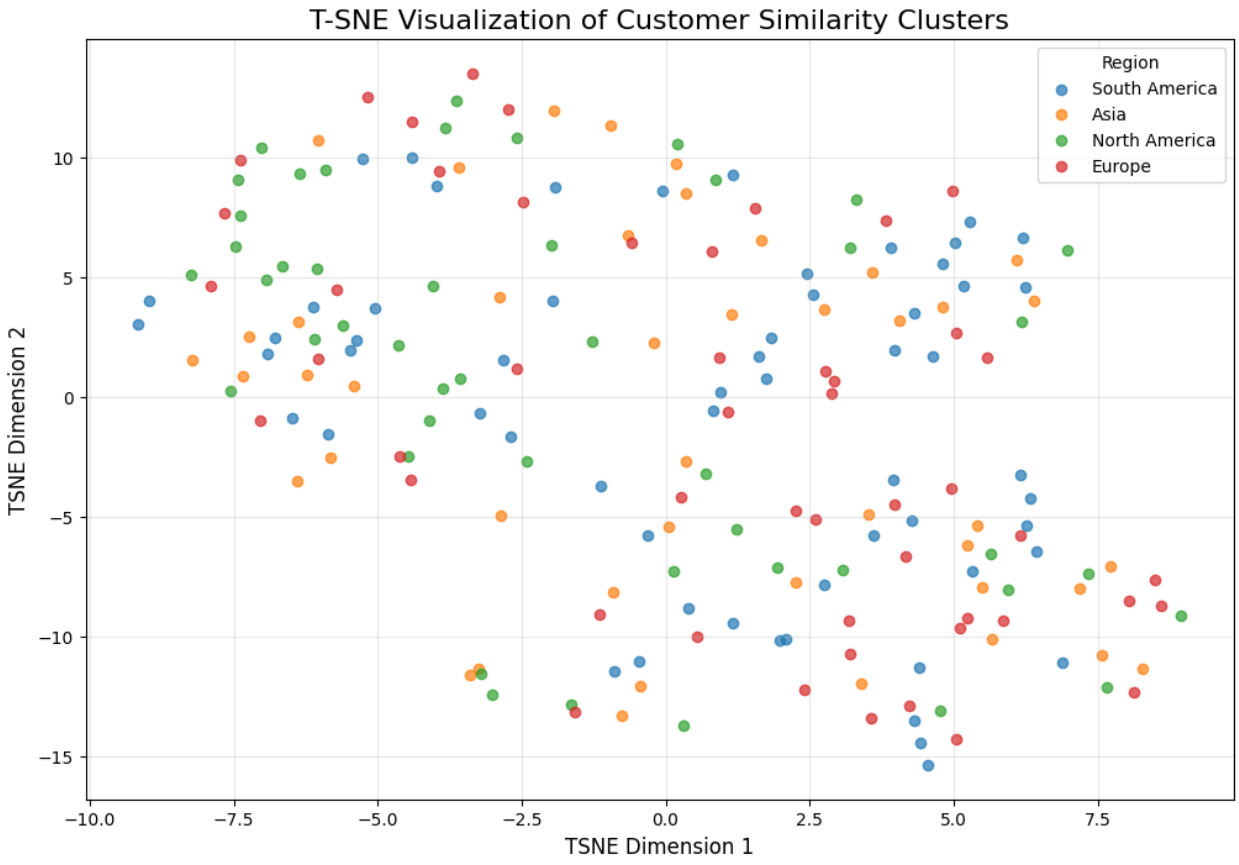
```
similarities if customer_ids[j] != customer_id][:3]
    recommendations_for_20[customer_id] = top_3

lookalike_map = [{"cust_id": cust, "lookalikes":
recommendations_for_20[cust]} for cust in recommendations_for_20]

import csv

with open("Lookalike.csv", mode="w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["cust_id", "lookalikes"])
    for entry in lookalike_map:
        writer.writerow([entry["cust_id"], entry["lookalikes"]])

print("Top-3 recommendations for the first 20 customers saved to
'Lookalike.csv'.")
```

```
Top-3 recommendations for the first 20 customers saved to
'Lookalike.csv'.
```

Evaluation

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import numpy as np

embedding_columns = [col for col in customer_profiles.columns if
col.startswith("Embedding_")]
numerical_features = [
    "TotalValue",
    "TotalQuantity",
    "DaysSinceSignup",
    "DaysSinceLastTransaction",
    "PurchaseFrequency",
    "AverageSpendPerTransaction",
]

combined_features = np.hstack([
    customer_profiles[embedding_columns].values,
    customer_profiles[numerical_features].values
])

tsne = TSNE(n_components=2, random_state=42, perplexity=30)
reduced_features = tsne.fit_transform(combined_features)

customer_profiles["TSNE_X"] = reduced_features[:, 0]
customer_profiles["TSNE_Y"] = reduced_features[:, 1]

plt.figure(figsize=(12, 8))
```

```
for region in customer_profiles["Region"].unique():
    subset = customer_profiles[customer_profiles["Region"] == region]
    plt.scatter(
        subset["TSNE_X"],
        subset["TSNE_Y"],
        label=region,
        alpha=0.7
    )

plt.title("T-SNE Visualization of Customer Similarity Clusters",
fontsize=16)
plt.xlabel("TSNE Dimension 1", fontsize=12)
plt.ylabel("TSNE Dimension 2", fontsize=12)
plt.legend(title="Region")
plt.grid(alpha=0.3)
plt.show()

customer_profiles.to_csv("customer_profiles_with_tsne.csv",
index=False)
print("T-SNE visualization completed. Profiles saved to
'customer_profiles_with_tsne.csv'.")
```



```
T-SNE visualization completed. Profiles saved to
'customer_profiles_with_tsne.csv'.
```